Detecting Nearby Devices in a Network Environment

Inventor(s):
Shell S. Simpson

EL792822375

DETECTING NEARBY DEVICES IN A NETWORK ENVIRONMENT

TECHNICAL FIELD

This invention relates to computer network topology determination, and
5   more particularly to detecting nearby devices in a network environment.

BACKGROUND

As computer technology has advanced the number of computers being
used, as well as the tasks these computers are being used to perform, has
10   increased. Accompanying this increased computer usage has been an increase
in the interconnectivity of computers. A wide variety of networking devices,
such as switches and hubs, have become increasingly commonplace in
businesses, educational facilities, and homes. These networking devices allow
data to be transferred among the computers that are coupled to the devices, as
15   well as to other devices that are coupled to the network, such as printers,
scanners, etc.

Although there are many benefits to interconnecting devices via a
network, there are also problems created by networking devices together. One
such problem is the wide variety of devices accessible to the user of computer.
20   For example, where the computer user once had a single printer attached to his
or her computer, he or she now has multiple networked printers to which he or
she can print. While this provides more options to the user, it can also make
things difficult for the user because he or she typically must identify which one
of the printers to print to. And, while the user typically desires to print to the
25   printer closest to him or her, there is often times little, if any, indication given
to the user as to which printer is closest.

Case No. 10007680-1

One solution to this problem is to have devices named in such a way that their names reflect their locations. For example, printers may be given names that indicate which floor of the building they are on, whose office they are close to, and so on. This solution, however, also has problems which can make

5   is user-unfriendly. Those problems include: the subjective nature of the device naming process, the need for human input into the device naming process, the need to rename a device if the device is moved, and the need for a user of a computing device to understand locations reflected in the names (for example, if the device name reflects that the device is close to Bob's office, then the user

10   needs to know the location of Bob's office).

Thus, a need exists for a more user-friendly way in which to identify devices on a network that are nearby a particular computer (or other device).

SUMMARY

15   Detecting nearby devices in a networked environment is described herein.

For each of one or more devices in a network, an inferred proximity of the device to another device (for example, a computing device) is determined based at least in part on which port of a network switch the device and the other

20   device are coupled to.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates an exemplary network environment in which nearby devices can be detected.

25   Fig. 2 is a more detailed block diagram illustrating a portion of networking environment 100 of Fig. 1.

Fig. 3 is a flowchart illustrating an exemplary process for detecting nearby devices in a network environment.

Fig. 4 is a flowchart illustrating an exemplary process for determining, based on the recorded switch and port information, the relative proximity of multiple desired devices.

Fig. 5 is a block diagram illustrating an exemplary computing device that detects desired nearby devices.

Fig. 6 illustrates an exemplary computer system in additional detail.

DETAILED DESCRIPTION

Detection of nearby devices in a network environment is described herein. The detection of devices near to a particular computing device is accomplished by discovering one or more switches that the computing device is coupled to. Information is obtained from these switches identifying devices that are coupled to particular ports of each switch. This information is then used to infer how near a device is to the computing device.

The discussions herein make reference to both network hubs and network switches (also commonly referred to as simply "hubs" and "switches"). Hubs and switches are both networking devices having multiple ports and that route data among the various devices coupled to the hub or switch by transmitting data out on one port (or ports) that has been received by the hub or switch on another port. The primary difference between a hub and a switch is the manner in which they identify which one or more ports to transmit data out of. A switch maintains a record of which devices are coupled to each of the switch's ports. Thus, when data is received that targets a particular device, the switch identifies which port the targeted device is coupled to and transmits the

3   *Case No. 10007680-1*

received data out on that identified port (but does not transmit the received data out on any other of the switch's ports). A hub, however, maintains no such record. Rather, any data received by the hub on a particular port is transmitted out on all other ports of the hub without regard for where the targeted device is located.

Fig. 1 illustrates an exemplary network environment 100 in which nearby devices can be detected. In network environment 100, an office 102 includes multiple computing devices (a desktop computer 104 and a portable computer 106) and a printer 108 coupled to a hub 110. The hub 110 is coupled to a switch 112, which is further coupled to another printer 114, another hub 116, another hub 118 that is coupled to a printer 120 in a conference room 122, yet another printer 124, and a central switch 126 that is further coupled to another switch 128. Many additional devices may also be included in network environment 100, however, these have not been shown in order to avoid cluttering the drawings.

Network environment 100 is organized in a hierarchical manner with networking devices located at multiple different hierarchical levels. Hubs 110,116, and 118 are situated at the lowest level, while floor switches 112 and 128 are situated at the next higher level. A top-level switch 126 is at the highest level and, although not shown, one or more additional levels (including hubs and/or switches) may also exist between switch 126 and switches 112 and 128. In the example environment 100, the network is implemented with one floor switch 112, 128 for each floor of a building. These floor switches 112, 128 are coupled to the central switch 126 (optionally via one or more additional hubs and/or switches).

4

The user of computer 104 or 106 will typically want to print to printer 108 because printer 108 is located in the same office as the user. However, the user may also take portable computer 106 to conference room 122, plugging in (or otherwise connecting) computer 106 to hub 118. If the user desires to print

5   from computer 106 while it is located in conference room 122, the user will typically want to print to the closest printer, which is printer 120.

In order to identify which device in network environment 100 to use (for example, which printer to print to), a device detection process is employed to infer how close one or more of the devices in network environment 100 is to

10   the computing device desiring to use the device. For example, while portable computer 106 is located office 102, the device detection process indicates that the printer 108 is closer to the computer 106 than printer 114, printer 120, or printer 124. However, when portable computer 106 is located in conference room 122, the device detection process indicates that the printer 120 is closer to

15   computer 106 than printer 108, printer 114, or printer 124.

Network environment 100 can be implemented in any of a wide variety of conventional manners. Couplings between devices can be wired (e.g., coax cable, fiber-optic cable, twisted pair, etc.) or wireless (e.g., RF, infrared, etc.), and can use any of a wide variety of conventional networking protocols

20   (e.g.,TCP/IP, NetBEUI, IPX/SPX, etc.).

Fig. 2 is a more detailed block diagram illustrating a portion of networking environment 100 of Fig. 1. A switch 160 is illustrated coupled to a printer 162 and a hub 164, which is further coupled to a computing device 166 and another printer 168. The components of Fig. 2 represent a variety of

25   different components in Fig. 1. For example: switch 160 can be floor switch 112 of Fig. 1, hub 164 can be hub 110 of Fig. 1, computing device 166 can be

5   *Case No. 10007680-1*

one of devices 104 or 106 of Fig. 1, printer 168 can be printer 108 of Fig. 1, and printer 162 can be printer 114 or 124 of Fig. 1. Each device 162, 166, and 168 includes a network interface 170, 172, and 174, respectively, that allows the device to communicate with other devices in the network. Each of the

5   interfaces 170, 172, and 174 further includes a network ID 176, 178, and 180, respectively, that uniquely identifies the corresponding network interface and thus the corresponding device on the network. Any of a wide variety of conventional identifiers can be used as network ID 176, 178, and 180, such as a hardware or MAC address, IP address, etc.

10   Switch 160 includes a switching fabric 182 and a connection table 184. Switching fabric 182 has multiple connection points or ports, one of which is connected to network interface 170, another of which is connected to hub 164, and unconnected ports are illustrated with terminating circles. Switching fabric 182 is responsible for receiving incoming network communications (e.g., data

15   packets) from a connected device and routing them appropriately. Connection table 184 maintains a record of which devices are coupled to switch 160. The devices are identified in table 184 by their respective network ID's, which are stored as entries 186 in table 184.

Switching fabric 182 uses the information stored in connection table 184

20   to perform the routing of received data. For example, if computing device 166 desires to print to printer 168, a print request (which identifies the network ID of printer 168 as the target device) is communicated from computing device 166 to hub 164 which in turn communicates the request out on all other ports of hub 164. The print request is thus received by printer 168 as well as switching

25   fabric 182. Switching fabric 182, based on connection table 184, determines that the network ID of the targeted device (network ID 180 of printer 168) is

6   *Case No. 10007680-1*

connected to the same port of switching fabric 182 as the print request was received on, and thus that the print request need not be transmitted out on any other port of fabric 182. However, if computing device 166 desires to print to printer 162, a print request is communicated from computing device 166 to hub

5   164 which in turn communicates the print request out on all other ports of hub 164. The print request is thus received by printer 168 (which ignores the request as it is not the targeted device) as well as switching fabric 182. Switching fabric 182, based on connection table 184, determines that the network ID of targeted device (network ID 176 of printer 162) is connected to a

10 port 188 of switching fabric 182. Thus, switching fabric 182 communicates the received print request to printer 162 by transmitting the print request out on port 188 (but not on any other port of fabric 182).

Fig. 3 is a flowchart illustrating an exemplary process 220 for detecting nearby devices in a network environment. The process of Fig. 3 is carried out

15 by a computing device in the network environment, typically the computing device for which detection of nearby devices is desired. In one implementation, the process of Fig. 3 is performed in software. The process of Fig. 3 can be carried out at different times, such as at preset (or random) intervals, when the device is powered on or reset, when the device is initially

20 connected to a network, when the network device that the computing device is directly connected to is changed (e.g., the computing device is moved and connected to a different hub), at the user's request, and so forth.

Initially, the switches in the network are discovered (act 222). The switches in the network can be discovered using any of a variety of well-known

25 discovery protocols. For example, SNMP (simple network management protocol) provides a mechanism by which devices possessing an SNMP agent

7   *Case No. 10007680-1*

(which switches usually do) can be discovered. After the switches in the network have been discovered, the devices of the desired type on the network are identified (act 224). Any of variety of different types of devices can be detected by the process of Fig. 3, such as printers, scanners, and so forth. In one implementation, the device performing the process of Fig. 3 has access to a list of network ID's (or alternatively other identifiers) of the desired devices in the network (for example, from a file stored at the device or alternatively elsewhere in the network). In another implementation, the device performing the process of Fig. 3 queries multiple addresses on the network (e.g., each address in the same subnet as the device's address). This querying process identifies both whether a device currently exists on the network with a particular address as well as the type of device at that address.

Once the switches and desired devices have been detected, each switch as well as which port of each switch that each desired device is coupled to is recorded (act 226). The switch and port information for each desired device is obtained from the connection tables of the switches in the network. Additionally, each switch and which port of each switch that the current device (the device for which the nearby desired devices are being detected) is coupled to is also obtained from the connection tables of the switches in the network and recorded (act 228). The switch and port information for a device can be obtained from the switches in any of a variety of manners, such as using the protocol described in Internet Engineering Task Force's (IETF) Request for Comments (RFC) 1493, entitled "Definition of Managed Objects for Bridges". Additional information regarding switch detection and device selection can be found in co-pending U.S. Patent Application serial number 09/730,074, filed

December 4, 2000, entitled "Selection of a Candidate Peripheral Device", to inventor Shell S. Simpson, which is hereby incorporated by reference.

Using this recorded switch and port information, the desired devices are ranked based on their inferred proximity to the computing device (act 230).

5 Inference is made as to the proximity of a desired device to the current device based on which port of the switch the devices are coupled to as well as how many other devices (whether or not of the same type as the desired device) are also coupled to that port. The rankings are then used as a basis for selection of one of the desired devices (act 232). This can be an automated selection (e.g.,

10 automatically select the closest printer (or randomly select one of the closest printers if there is more than one) to the user as his or default printer), or alternatively the rankings may be presented to the user to allow him or her to select one of the devices.

Fig. 4 is a flowchart illustrating an exemplary process 240 for

15 determining, based on the recorded switch and port information, the relative proximity of multiple desired devices. The process of Fig. 4 is carried out by a computing device in the network environment, typically the computing device for which detection of nearby devices is desired. In one implementation, the process of Fig. 4 is performed in software. The process of Fig. 4 ranks devices

20 into one of up to five different categories, with devices in a lower number category being closer to the computing device than devices in a higher number category (with category 1 being the closest devices and category 5 being the furthest devices). Situations can arise where a desired device can be ranked in multiple different categories. In such situations, the desired device is ranked in

25 the lowest numbered category of those multiple categories.

9

Initially, the discovered switch and port information for the devices is received (act 242). This includes the switch and port information for the desired device(s) from act 226 of Fig. 3 as well as the switch and port information for the current device from act 228 of Fig. 3. One or more of the following acts (act 244-act 260) are performed for each desired device. A check is made as to whether the desired device shares the same port on the switch as the current device and with the smallest number of other devices also sharing that port (act 244). If so, the device is determined to be one of the closest devices and is ranked in category 1 (act 246). If not, a check is made as to whether the desired device shares the same port on the switch as the current device and with a smaller number of other devices also sharing that same port (act 248). If so, the device is determined to be one of a set of one or more second closest devices and is ranked in category 2 (act 250). If not, a check is made as to whether the desired device shares the same port on the switch as the current device without regard for the number of other devices also sharing that port (act 252). If so, the device is determined to be one of a set of one or more third closest devices and is ranked in category 3 (act 254). If not, a check is made as to whether the desired device shares the same switch as the current device with any number of other devices also sharing that same switch (act 256). If so, the device is determined to be one of a set of one or more fourth closest devices and is ranked in category 4 (act 258), and if not the device is determined to be one of a set of one or more fifth closest devices (or furthest devices) and is ranked in category 5 (act 260). It is to be appreciated that the discovery of switch and port information may not always be a perfect process (e.g., a root switch might not be discovered or might not allow the client to access its SNMP agent), and thus act 260 is included in process 240.

10

Alternatively, in embodiments where such uncertainty or imperfection does not exist, process 240 may not include act 260.

The process of Fig. 4 can be explained by way of example with reference to network environment 100 of Fig. 1. Assume that the desktop
5   computer 104 in office 102 of Fig. 1 desires to detect nearby printers and employs the process of Fig. 4. Printer 108 shares the same port on switch 112 as computer 104, so printer 108 shares the same port on switch 112 with the smallest number of other devices also sharing that port and thus is ranked in category 1. This is so even though printer 108 also shares the same port on
10   switch 126 as computer 104 and various other devices (because the number of other devices sharing that port on switch 126 is larger).

Printers 114, 120, and 124 do not share a port of switch 112 with computer 104, however they do share a port of switch 126 with computer 104. Thus, printers 114, 120, and 124 are ranked in category 2. Alternatively, if
15   there are additional levels of switches between switch 112 and switch 126, then printers 114, 120, and 124 share a port of the switch at the next higher level than switch 112 and are ranked at category 2 based on this switch. Any printers (not shown) connected to switch 128 would share the same switch 126 as computer 104 (but not the same port on switch 126), and thus would be
20   ranked in category 4.

It should be noted that the acts of the processes of Figs. 3 and 4 can be implemented in different orders than are illustrated in the Figures, and that no required ordering should be implied based on the ordering of the illustrated acts. For example, in Fig. 3 act 226 could be performed prior to, subsequent to,
25   or concurrent with act 228.

11

Fig. 5 is a block diagram illustrating an exemplary computing device 280 that detects desired nearby devices. Computing device 280 includes a network interface 282 analogous to network interfaces 170, 172, and 174 of Fig. 2, via which computing device 280 can communicate with other computing devices via one or more networking devices. Computing device 280 also includes a device proximity detector 284 having a switch identification module 286, a device identification module 288, and a proximity ranking module 290. Switch identification module 286 identifies switches in the network (e.g., performing act 222 of Fig. 3). Device identification module identifies desired devices in the network (e.g., performing act 224 of Fig. 3), and proximity ranking module 290 ranks the proximity of the various identified desired devices (e.g., performing acts 226 and 228 of Fig. 3, and the process 240 of Fig. 4).

Fig. 6 illustrates an exemplary computer system 300 in additional detail. Computer system 300 can be, for example, a desktop computer 104 or portable computer 106 of Fig. 1, a computing device 166 of Fig. 2, a device 280 of Fig. 5, etc. Computer system 300 represents a wide variety of computing devices, such as desktop computers, portable or handheld computers, dedicated server computers, multi-processor computing devices, cellular telephones, personal digital assistants (PDAs), handheld or pen-based computers, microcontroller-based electronic devices, gaming consoles, television set-top boxes, and so forth.

Computer system 300 includes a processor 302, a memory 304, a mass storage device 306 (e.g., a fixed or removable magnetic or optical disk), and an input/output (I/O) interface 308, all coupled to a bus 310. Bus 310 represents one or more buses in computer system 300, such as a system bus, processor

12   *Case No. 10007680-1*

bus, accelerated graphics port (AGP), peripheral component interconnect (PCI), and so forth. The bus architecture can vary by computing device as well as by manufacturer. I/O interface 308 is a conventional interface allowing components of system 300 (e.g., processor 302) to communicate with other

5   computing devices via a network. I/O interface 308 may be, for example, a modem, a network interface card (NIC), and so forth.

Memory 304 represents volatile and/or nonvolatile memory used to store instructions and data for use by processor 302. Typically, instructions are stored on a mass storage device 306 (or nonvolatile memory) and loaded into a

10   volatile memory 304 for execution by processor 302. Additional memory components may also be involved, such as cache memories internal or external to processor 302. Various embodiments of the invention may be implemented, at different times, in any of a variety of computer readable media that is part of, or readable by, system 300. For example, such computer readable media may

15   be mass storage device 306, memory 304 or a cache memory, and so forth.

Computer system 300 is exemplary only. It is to be appreciated that additional components (not shown) can be included in system 300 and some components illustrated in system 300 need not be included. For example, a display adapter, additional processors or storage devices, additional I/O

20   interfaces, and so forth may be included in system 300, or mass storage device 306 may not be included.

The discussions herein refer primarily to software components and modules that can be executed by a computing device. It is to be appreciated, however, that the components and processes described herein can be

25   implemented in software, firmware, hardware, or a combination thereof. By way of example, a programmable logic device (PLD) or application specific

13   *Case No. 10007680-1*

integrated circuit (ASIC) could be configured or designed to implement various components and/or processes discussed herein.

Although the description above uses language that is specific to structural features and/or methodological acts, it is to be understood that the
5   invention defined in the appended claims is not limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the invention.

Case No. 10007680-1